

## Statistics: Covariance



## Covariance

- Measures linear relationship between two variables

## Statistics: Covariance

- Covariance (Population)

$$\text{Cov}(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n}$$

Where

$\Sigma$  = the sum of

$X_i$  = individual datum value

$\bar{X}$  = population mean

$n$  = the number of datum in the population

## Statistics: Covariance

- Covariance (Sample)

$$\text{Cov}(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n - 1}$$

Where

$\Sigma$  = the sum of

$X_i$  = individual datum value

$\bar{X}$  = sample mean

$n$  = the number of datum in the sample

## Statistics: Covariance

$$\text{Cov}(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n - 1}$$

X	Y	$\bar{X}$	$\bar{Y}$	$(X - \bar{X})$	$(Y - \bar{Y})$	$(X - \bar{X})(Y - \bar{Y})$
Temerature	Customer					
100	60	87.5	180	12.5	-120	-1495.83
95	98	87.5	180	7.5	-81.7	-612.5
90	100	87.5	180	2.5	-79.7	-199.167
85	200	87.5	180	-2.5	20.3	-50.8333
80	300	87.5	180	-7.5	120	-902.5
75	320	87.5	180	-12.5	140	-1754.17

$$\sum (X - \bar{X})(Y - \bar{Y}) = -5015$$

$$\sum (X - \bar{X})(Y - \bar{Y}) / n-1 = -5015 / 5$$

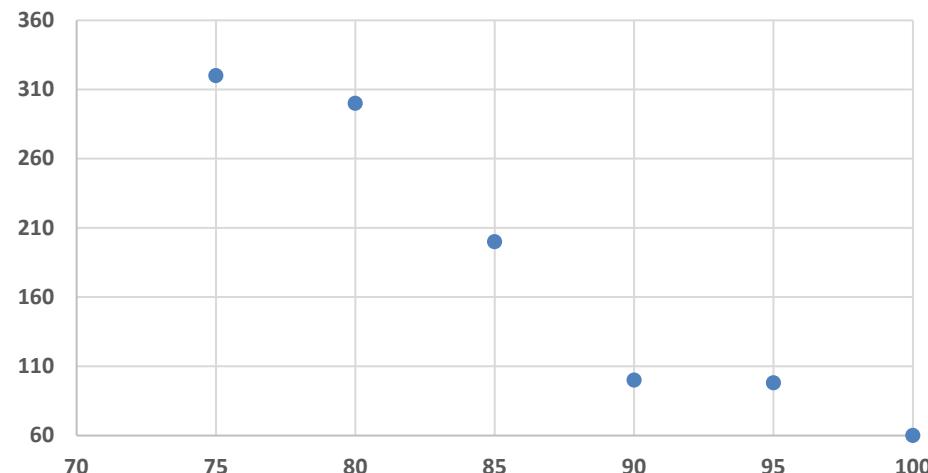
$$\text{Cov}(X, Y) = -1003$$

## Statistics: Covariance

X	Y
Temerature	Customer
100	60
95	98
90	100
85	200
80	300
75	320

$$\text{Cov}(X,Y) = -1003$$

Temerature vs. Customer

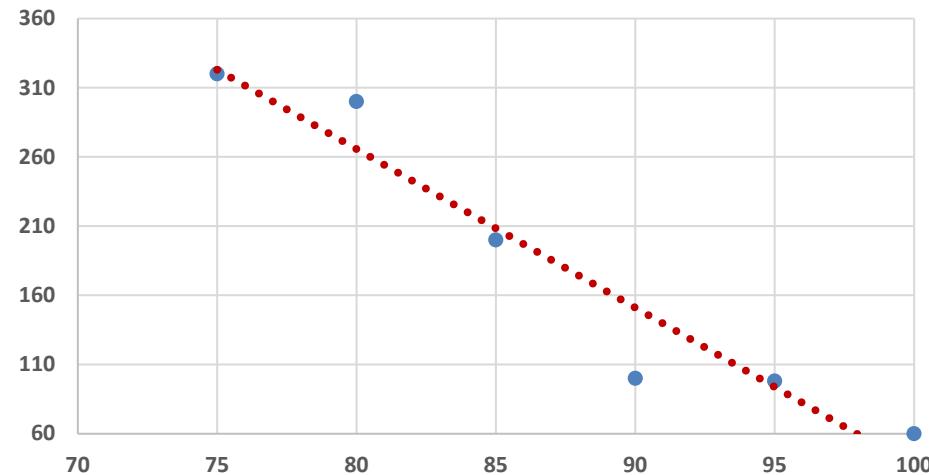


## Statistics: Covariance

X	Y
Temerature	Customer
100	60
95	98
90	100
85	200
80	300
75	320

$$\text{Cov}(X,Y) = -1003$$

Temerature vs. Customer



## Statistics: Covariance

X	Y	X	$\bar{y}$	(X-X)	(y- $\bar{y}$ )	(X-X)(y- $\bar{y}$ )
Chips	Weight					
2	120	4.83	185	-2.83	-65.3	185.111
5	207	4.83	185	0.17	21.7	3.61111
3	122	4.83	185	-1.83	-63.3	116.111
8	247	4.83	185	3.17	61.7	195.278
5	227	4.83	185	0.17	41.7	6.94444
6	189	4.83	185	1.17	3.67	4.27778

$$\sum (X-\bar{X})(Y-\bar{Y}) = 511.33$$

$$\sum (X-\bar{X})(Y-\bar{Y}) / n-1 = 511.33 / 5$$

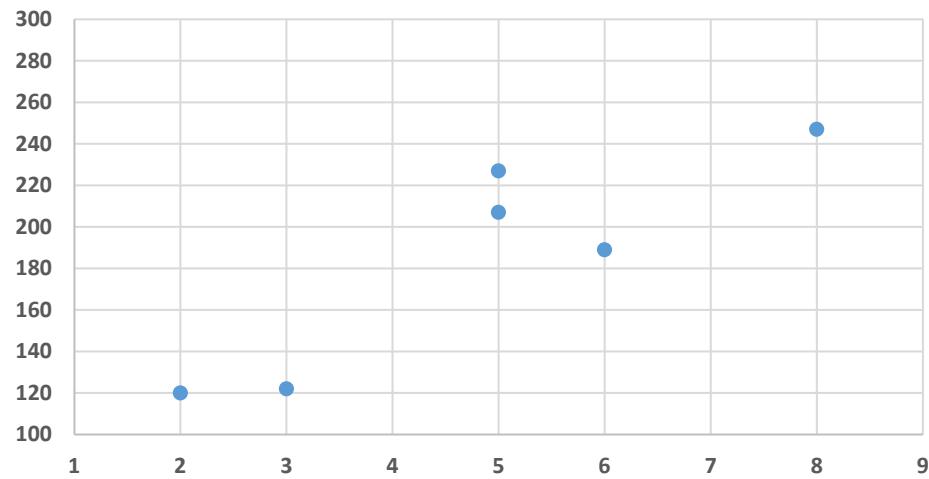
$$\text{Cov}(X,Y) = 102.26$$

## Statistics: Covariance

X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189

$$\text{Cov}(X,Y) = 102.26$$

Chips Vs. Weight

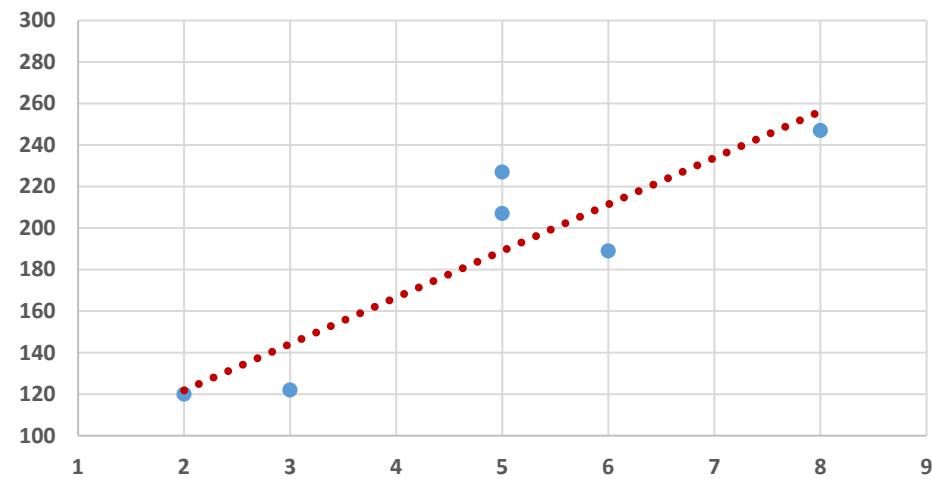


## Statistics: Covariance

X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189

$$\text{Cov}(X,Y) = 102.26$$

Chips Vs. Weight



Statistics: : Covariance

- **Cov(X,Y) = 0 ?**

## Statistics: : Covariance

- + means positive linear relationship
- - means negative linear relationship
- 0 means no linear relationship
- It only gives the direction of the relationship
- No indication of the strength of the relationship

## Statistics: Correlation



## Correlation

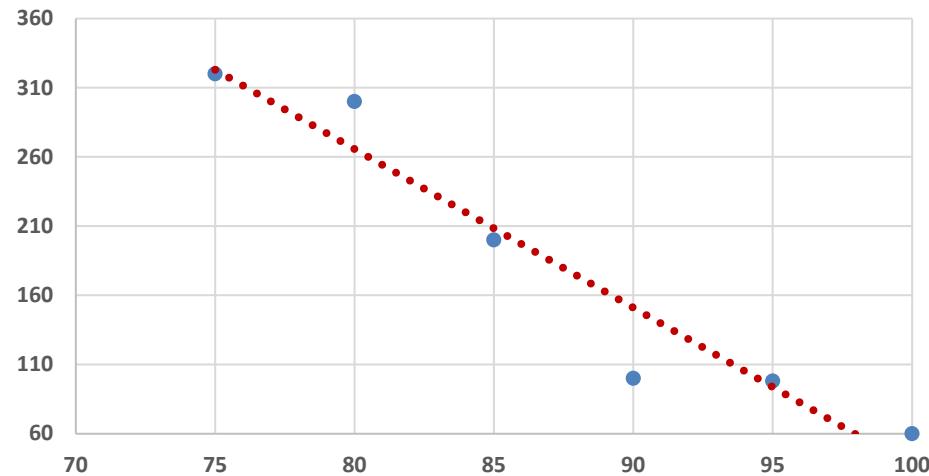
- Measures relationship and strength of the relationship between two variables

## Statistics: Covariance

X	Y
Temerature	Customer
100	60
95	98
90	100
85	200
80	300
75	320

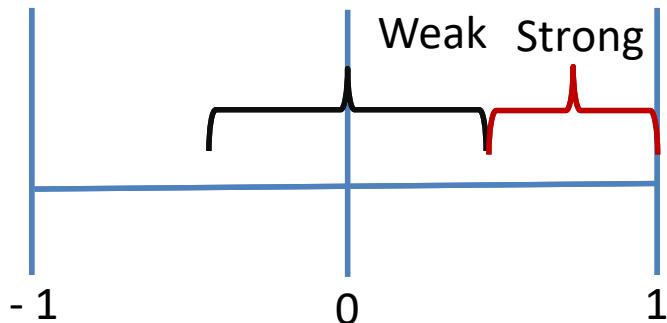
$$\text{Cov}(X,Y) = -1003$$

Temerature vs. Customer



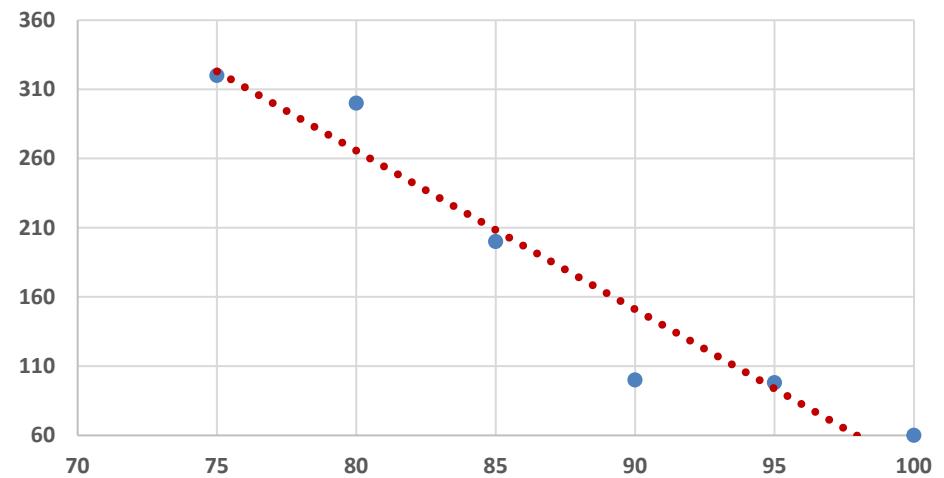
## Statistics: Correlation

- Pearson Correlation Coefficient
  - Pearson r



$$r = \text{Cov}(X,Y) / s_x s_y$$

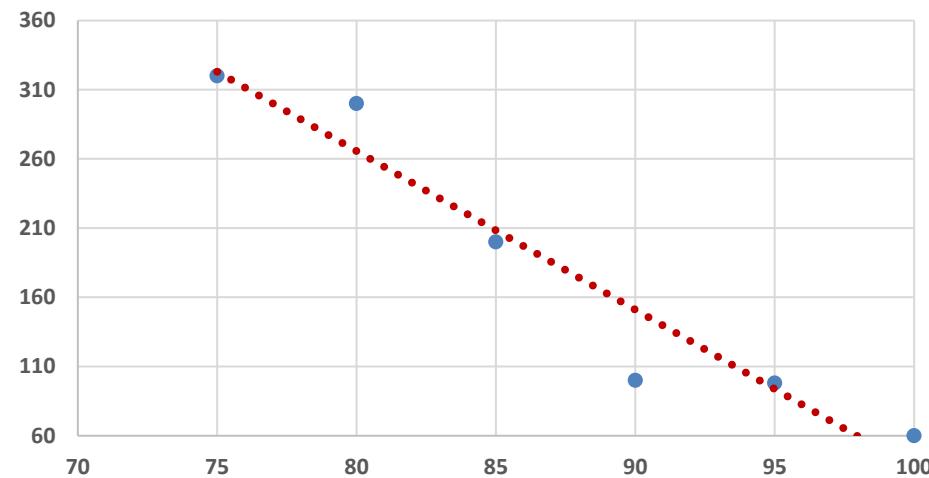
Temerature vs. Customer



## Statistics: Correlation

X	Y
Temerature	Customer
100	60
95	98
90	100
85	200
80	300
75	320

Temerature vs. Customer



$$\text{Cov}(X,Y) = -1003$$

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

$$S_x = 9.35$$

$$S_y = 111.29$$

$$r = \text{Cov}(X,Y) / s_x s_y = -1003 / (9.35 * 111.29)$$

$$r = -0.9635$$

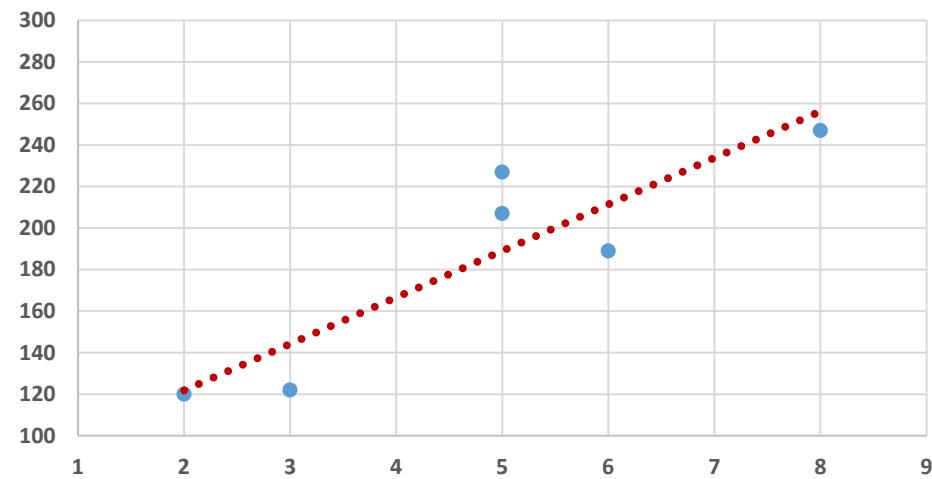
## Statistics: Correlation

X	Y
Chips	Weight
2	120
5	207
3	122
8	247
5	227
6	189

$$\text{Cov}(X,Y) = 102.26$$

$$r = \text{Cov}(X,Y) / s_x s_y = 0.8948$$

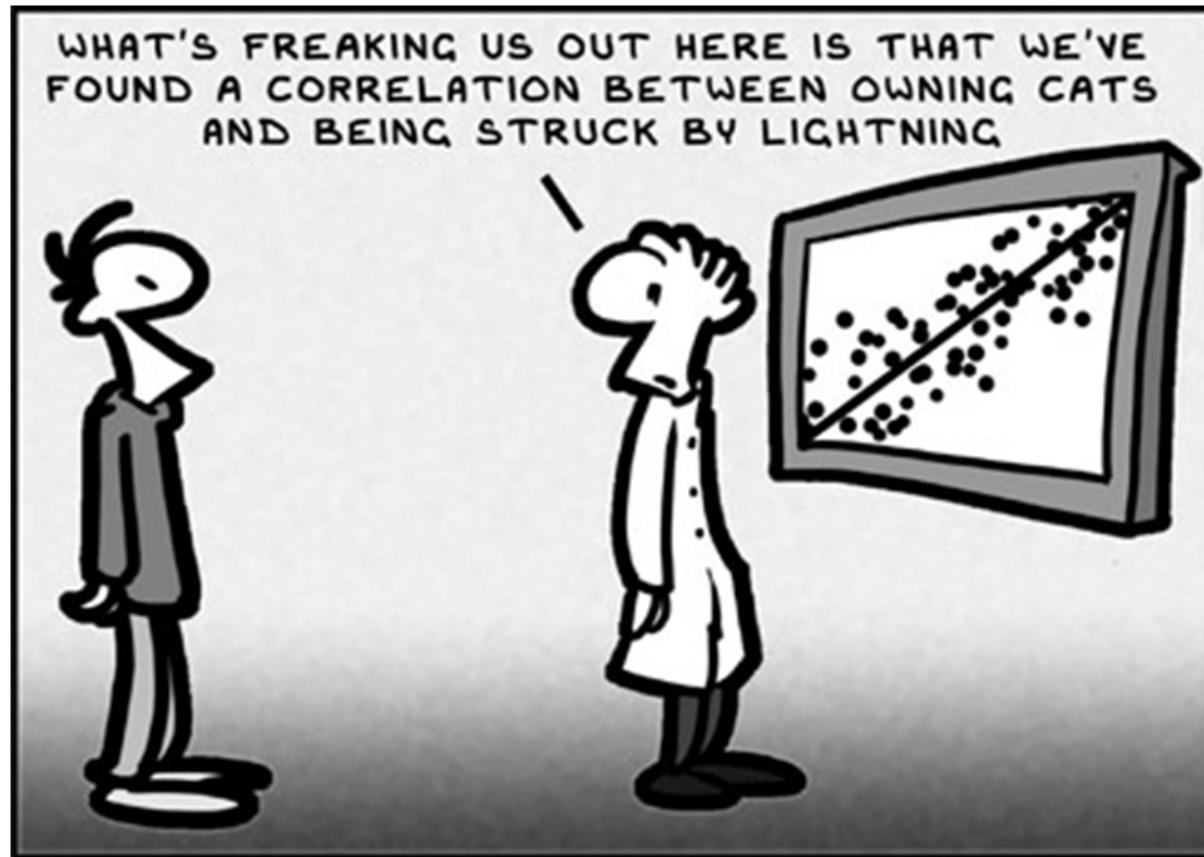
Chips Vs. Weight



## Statistics: Correlation

- Correlation  
does not equal  
to causation

## Statistics: Correlation



## Python Introduction and Installation



A programming language created by Guido van Rossum and was made available in 1991.

Python can be used for:

Web programs,  
Scripts,  
Math,  
Big data management,  
Data analytics

## Python Introduction and Installation

- Why Python?
- Portable
  - Windows, Mac, Linux, Raspberry Pi, etc.
- “common sense” syntax
  - Easy to learn
  - Simplicity
- Interpretive programming language
- Highly forgiven
  - procedural, functional, object-orientated

## Python Installation

- Anaconda
  - Python distribution and related libraries

## Python Installation Go to [Anaconda.com/download](https://anaconda.com/download)

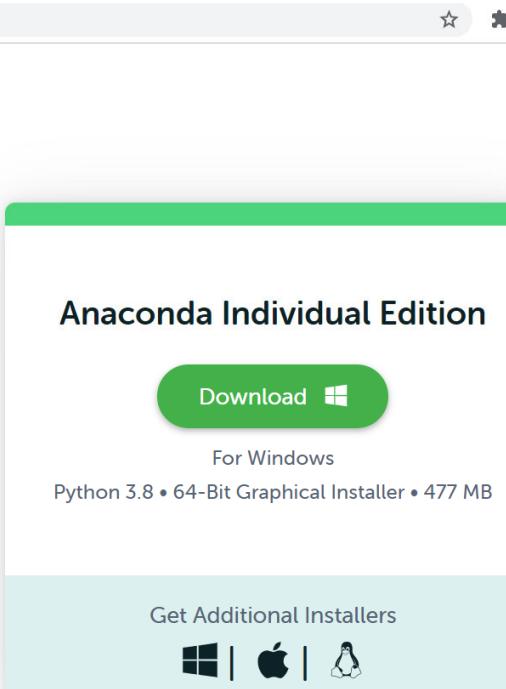
[anaconda.com/products/individual](https://anaconda.com/products/individual)



Individual Edition

# Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

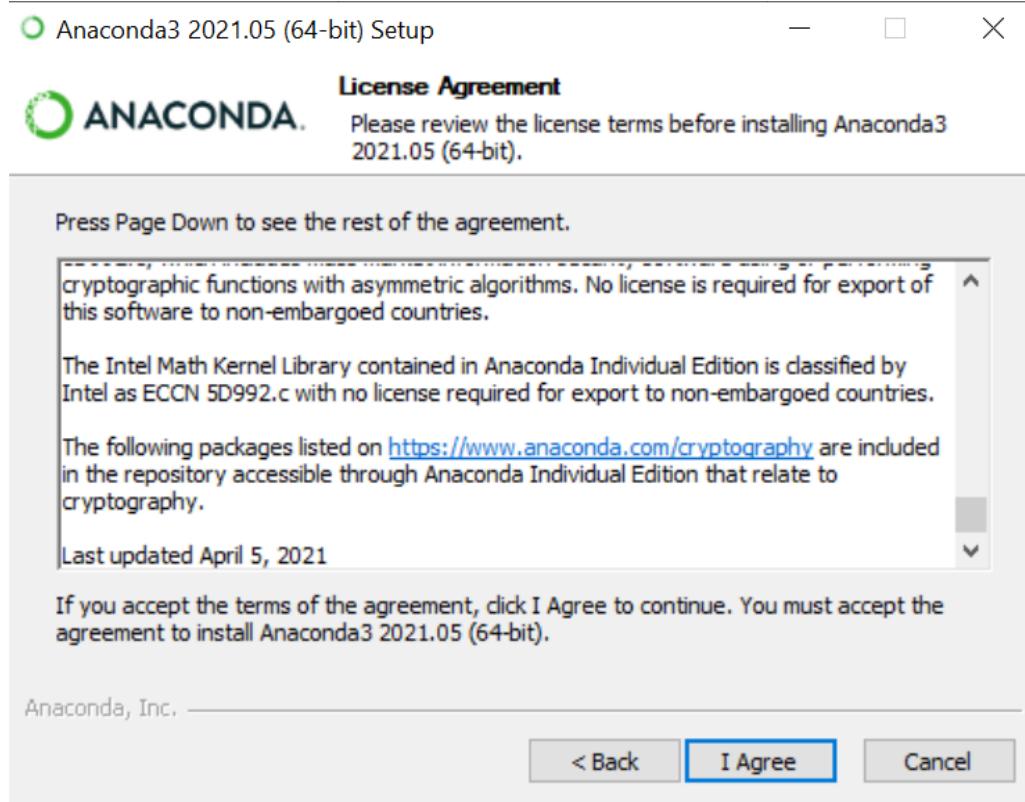


The screenshot shows the Anaconda Individual Edition download page. At the top, there's a green header bar. Below it, the text "Anaconda Individual Edition" is displayed. A prominent green button labeled "Download" with a Windows icon is centered. To its right, the text "For Windows" and "Python 3.8 • 64-Bit Graphical Installer • 477 MB" is shown. At the bottom, a light blue footer bar contains the text "Get Additional Installers" and icons for Windows, Apple, and Linux.

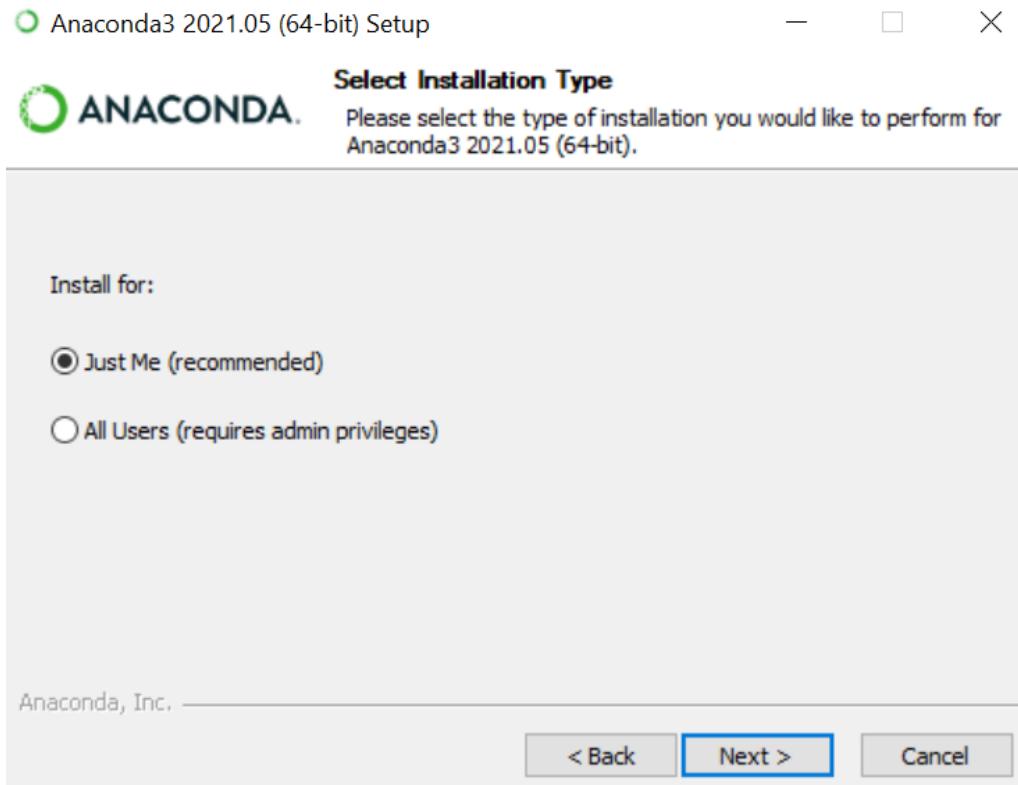
## Python Installation Go to [Anaconda.com/download](https://Anaconda.com/download)



# Python Installation



# Python Installation



# Python Installation

○ Anaconda3 2021.05 (64-bit) Setup



## Choose Install Location

Choose the folder in which to install Anaconda3 2021.05 (64-bit).

Setup will install Anaconda3 2021.05 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

G:\Thinkpad\bhuang\Anaconda\

[Browse...](#)

Space required: 2.9GB

Space available: 587.5GB

Anaconda, Inc. —

< Back

Next >

Cancel

# Python Installation

Anaconda3 2021.05 (64-bit) Setup



## Advanced Installation Options

Customize how Anaconda integrates with Windows

### Advanced Options

Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

Register Anaconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Anaconda, Inc. —

< Back

Install

Cancel

# Python Installation

○ Anaconda3 2021.05 (64-bit) Setup — □ ×



## Advanced Installation Options

Customize how Anaconda integrates with Windows

### Advanced Options

Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

Register Anaconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

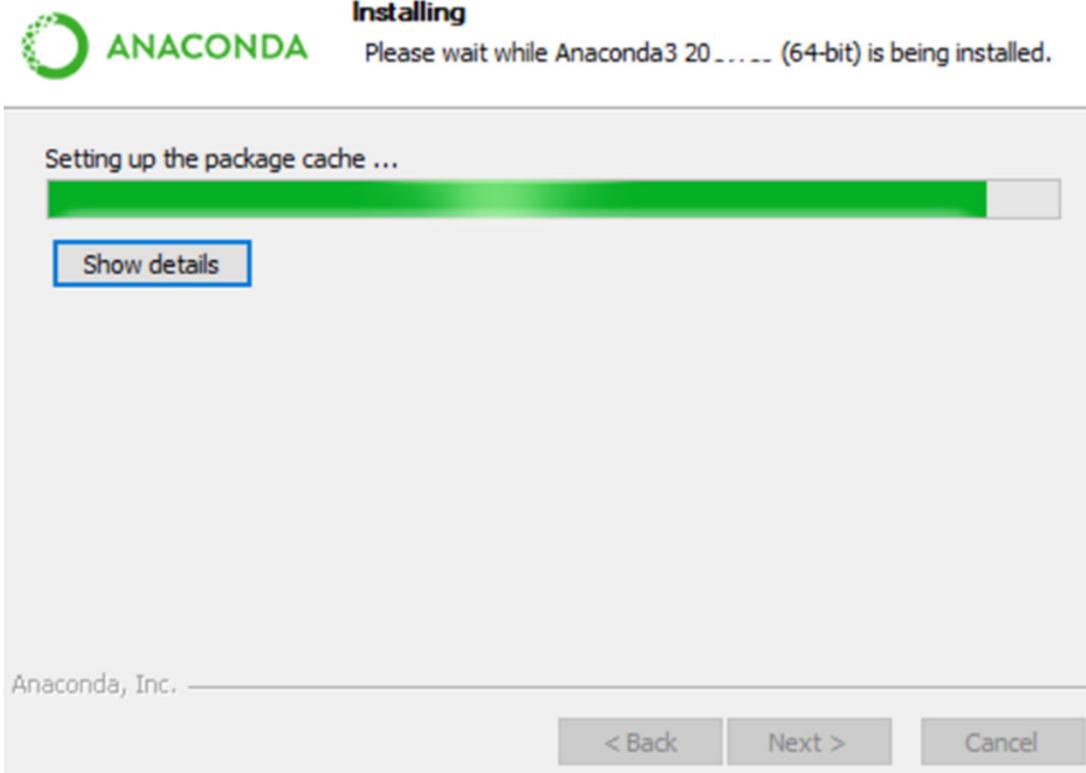
Anaconda, Inc. —

< Back

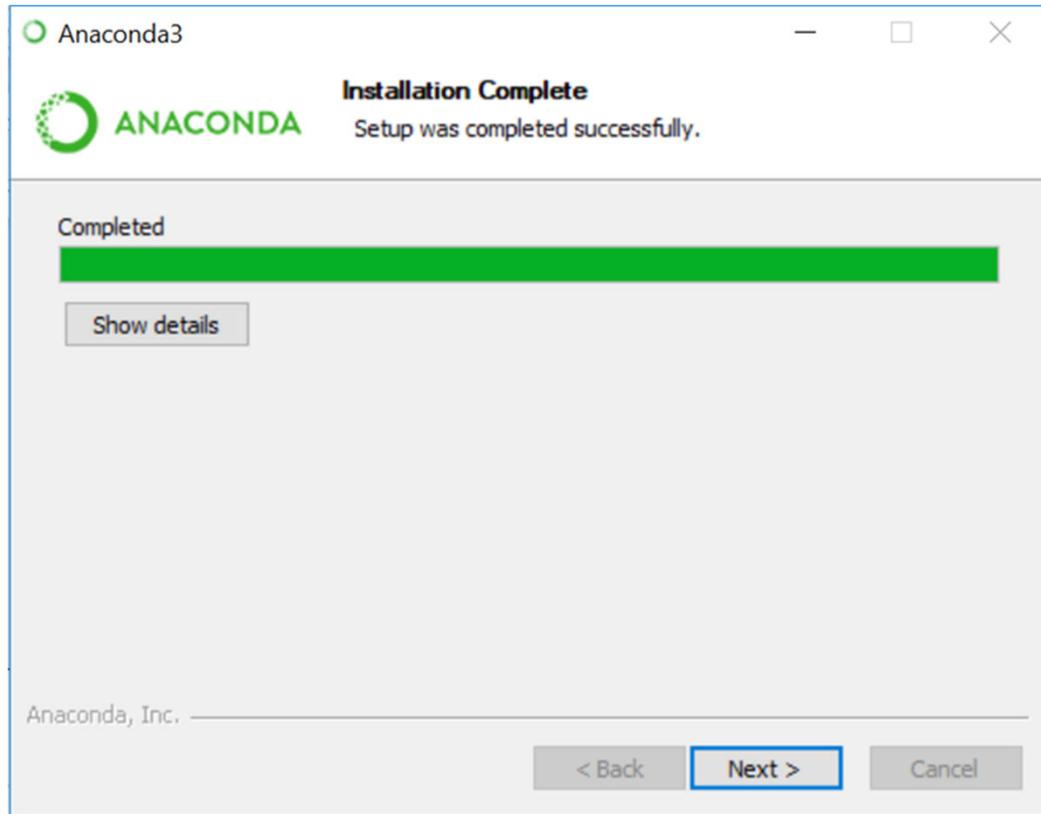
Install

Cancel

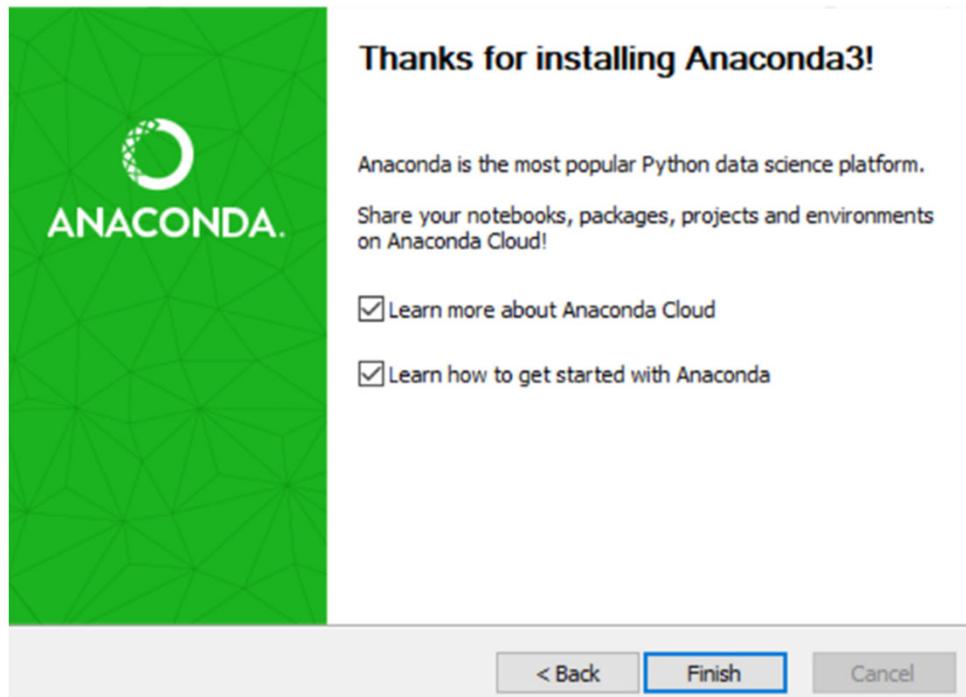
# Python Installation



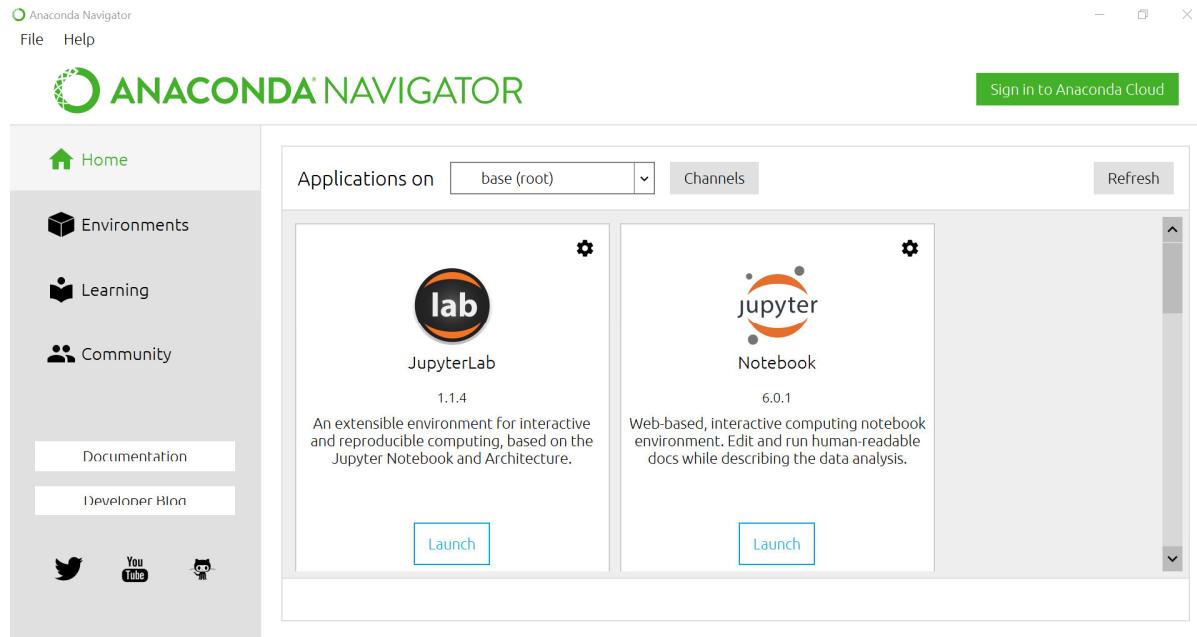
# Data Science: Python Installation



# Python Installation



# Python Installation



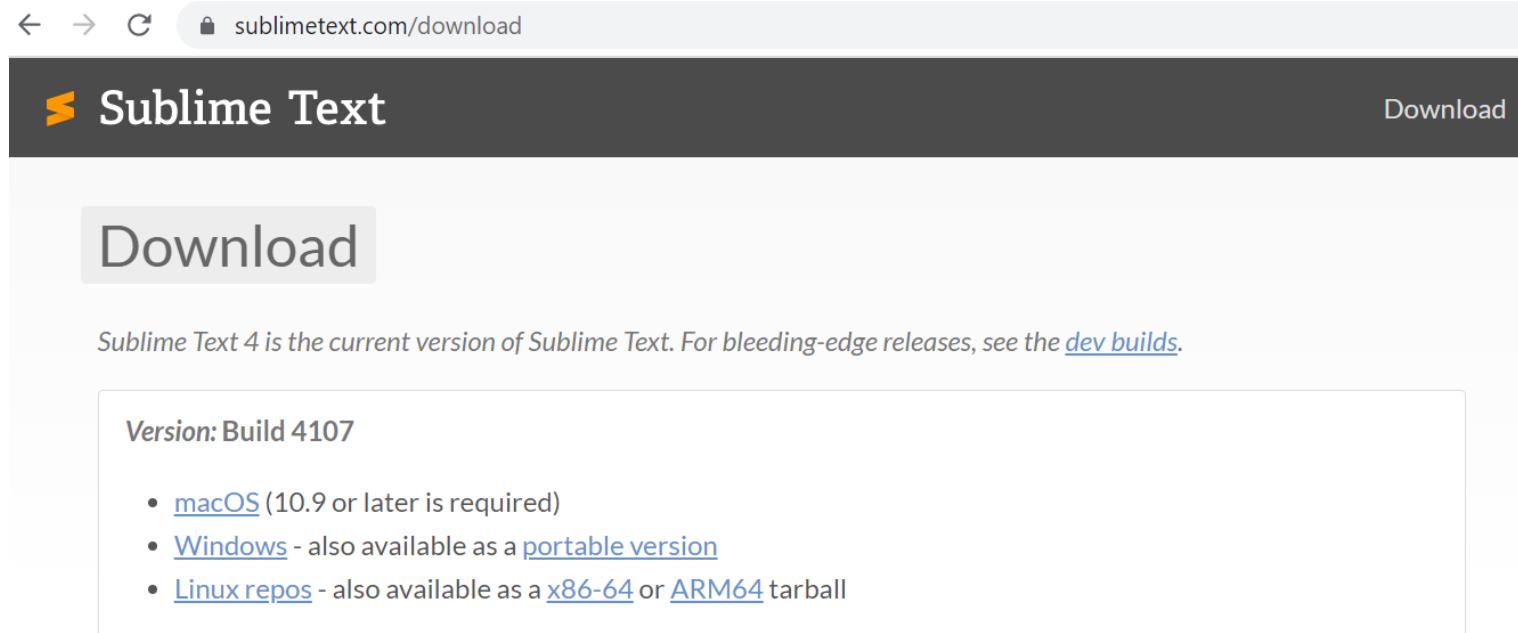
# Python Introduction and Installation



## Python Sublime Text Editor



Python Sublime Text Editor • Go to <https://www.sublimetext.com/download>



The screenshot shows a web browser window displaying the Sublime Text download page at <https://www.sublimetext.com/download>. The page has a dark header with the Sublime Text logo and a 'Download' button. Below the header, a large 'Download' button is prominently displayed. A note below it states: "Sublime Text 4 is the current version of Sublime Text. For bleeding-edge releases, see the [dev builds](#)". A section titled "Version: Build 4107" lists download links for macOS, Windows, and Linux.

← → ⌛ 🔒 sublimetext.com/download

# Sublime Text

Download

## Download

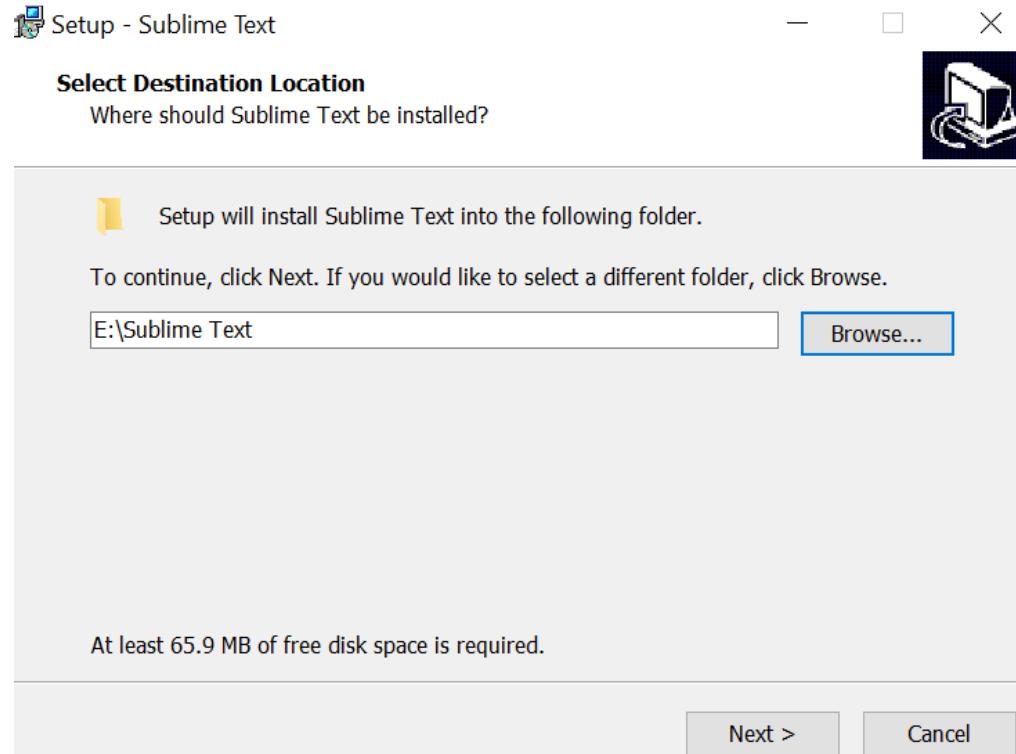
Sublime Text 4 is the current version of Sublime Text. For bleeding-edge releases, see the [dev builds](#).

Version: Build 4107

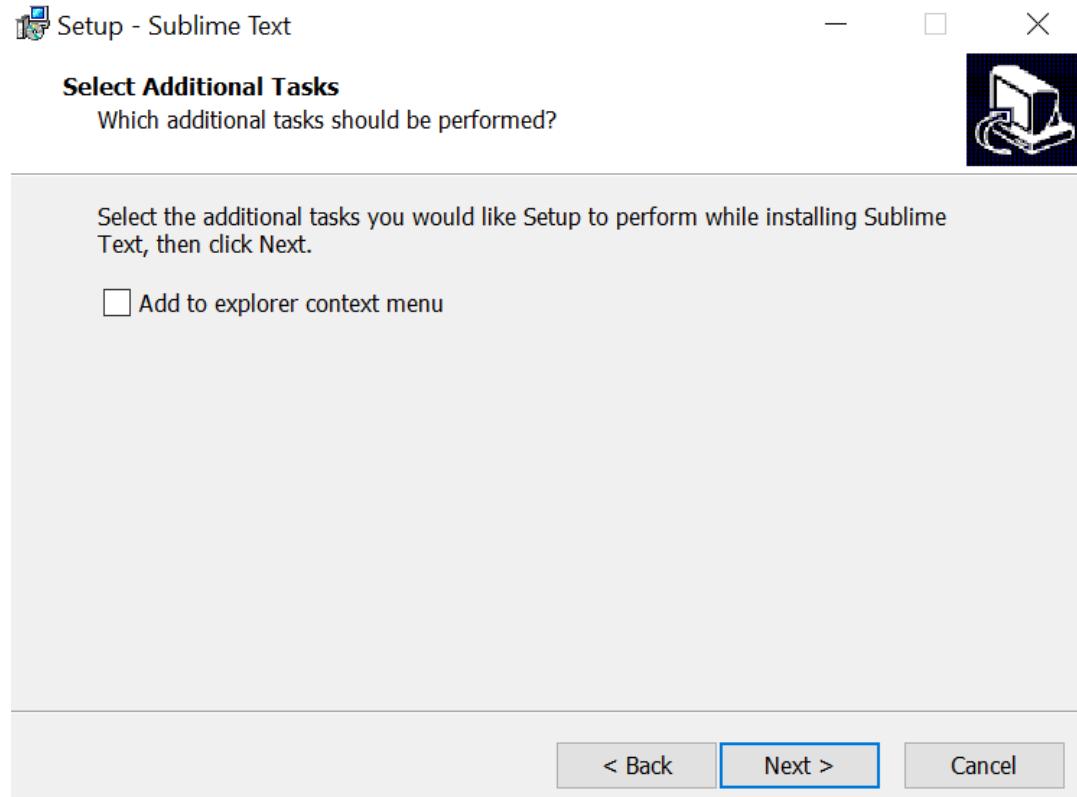
- [macOS](#) (10.9 or later is required)
- [Windows](#) - also available as a [portable version](#)
- [Linux repos](#) - also available as a [x86-64](#) or [ARM64](#) tarball

Sublime Text may be downloaded and evaluated for free, however a license must be [purchased](#) for continued use. There is currently no enforced time limit for the evaluation.

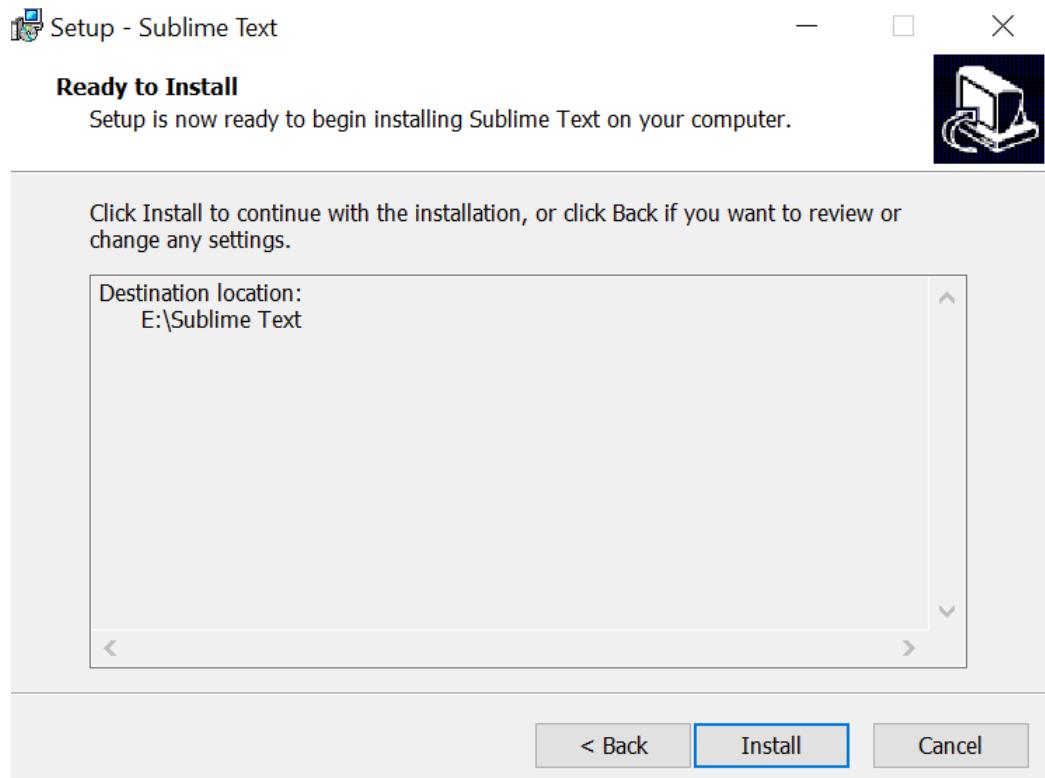
# Python Sublime Text Editor



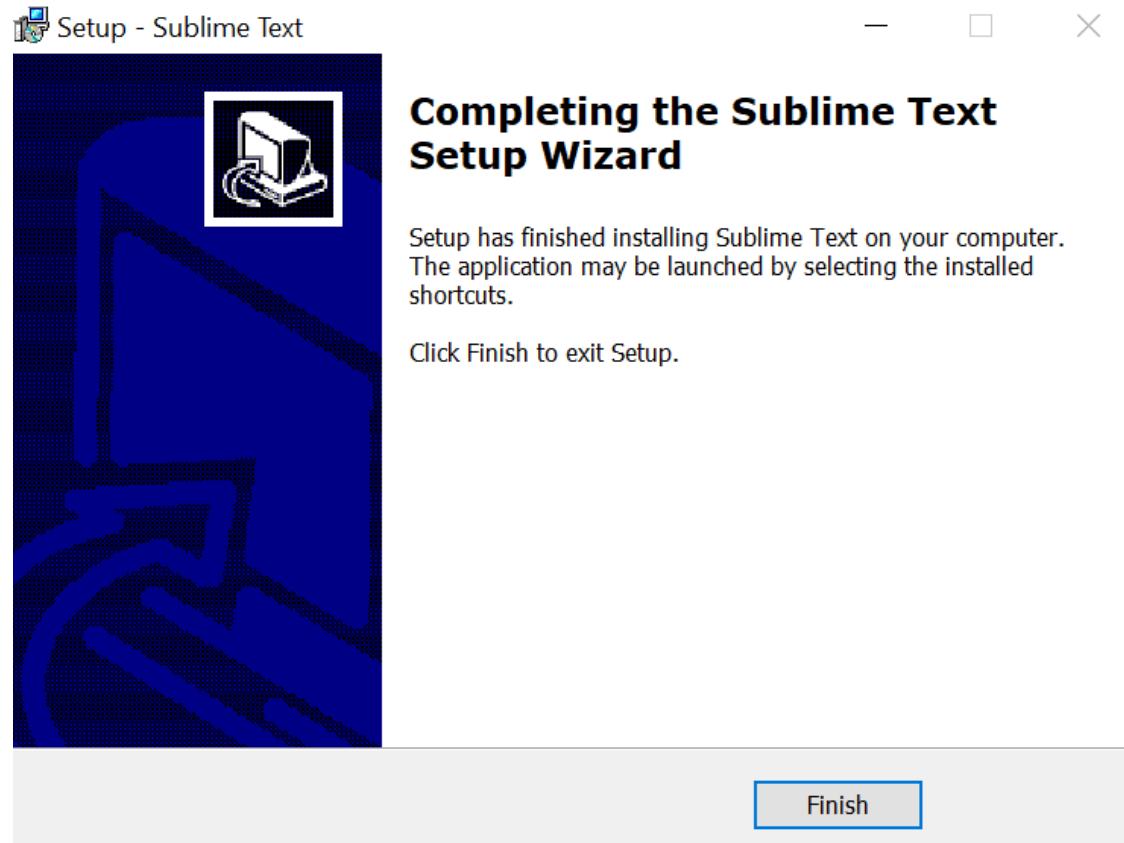
# Python Sublime Text Editor



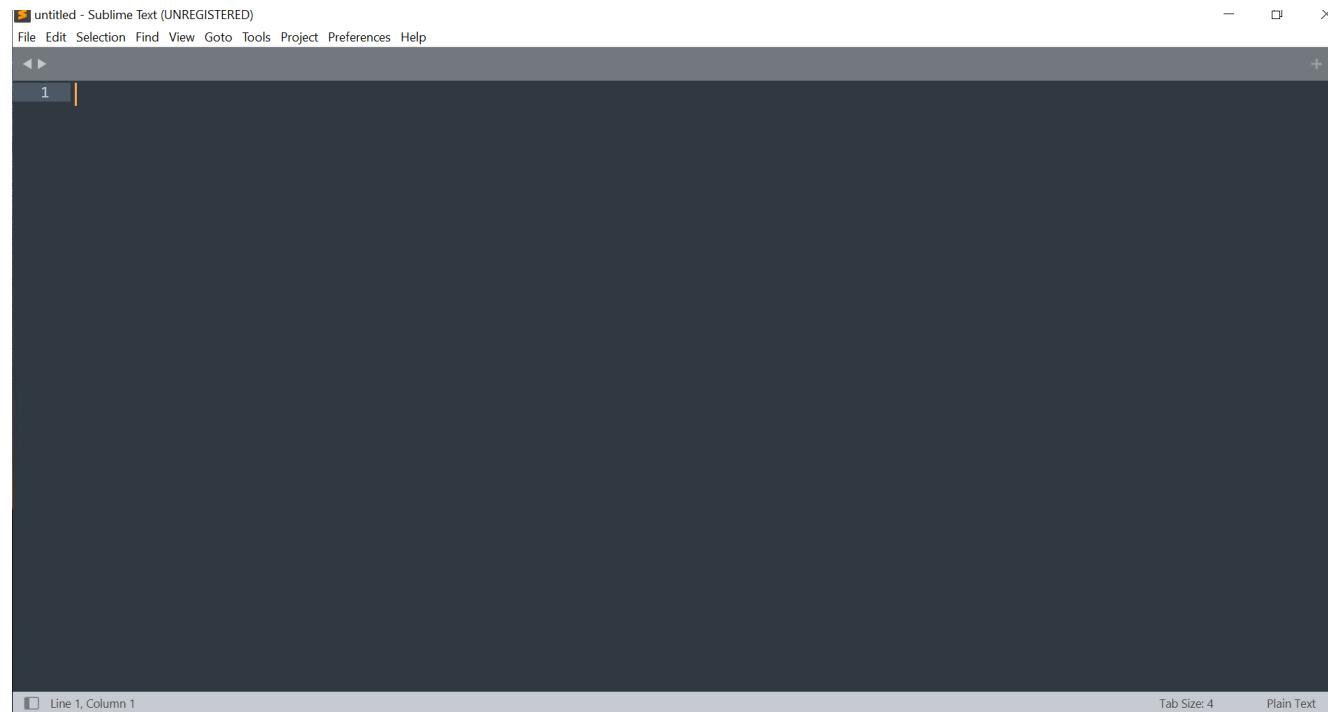
# Python Sublime Text Editor



# Python Sublime Text Editor



# Python Sublime Text Editor



## Data Science: Python Sublime Text Editor



## Data Science: Python Prompt via Command Prompt

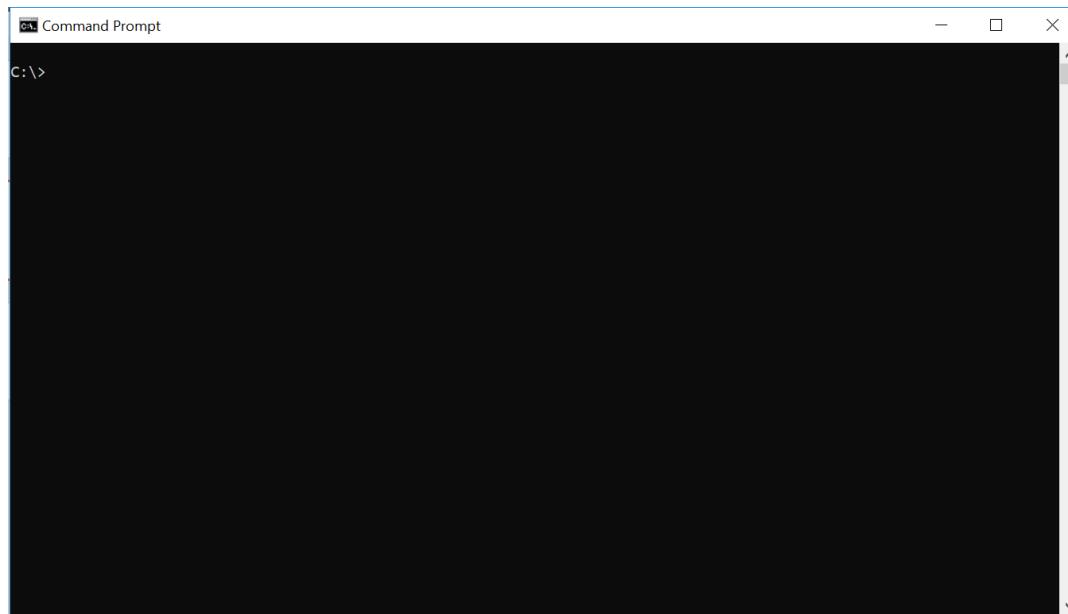


## Python Prompt via Command Prompt

To get to the Python Prompt:

### 1. Launch Command Prompt (The Mac OS equivalent is Terminal)

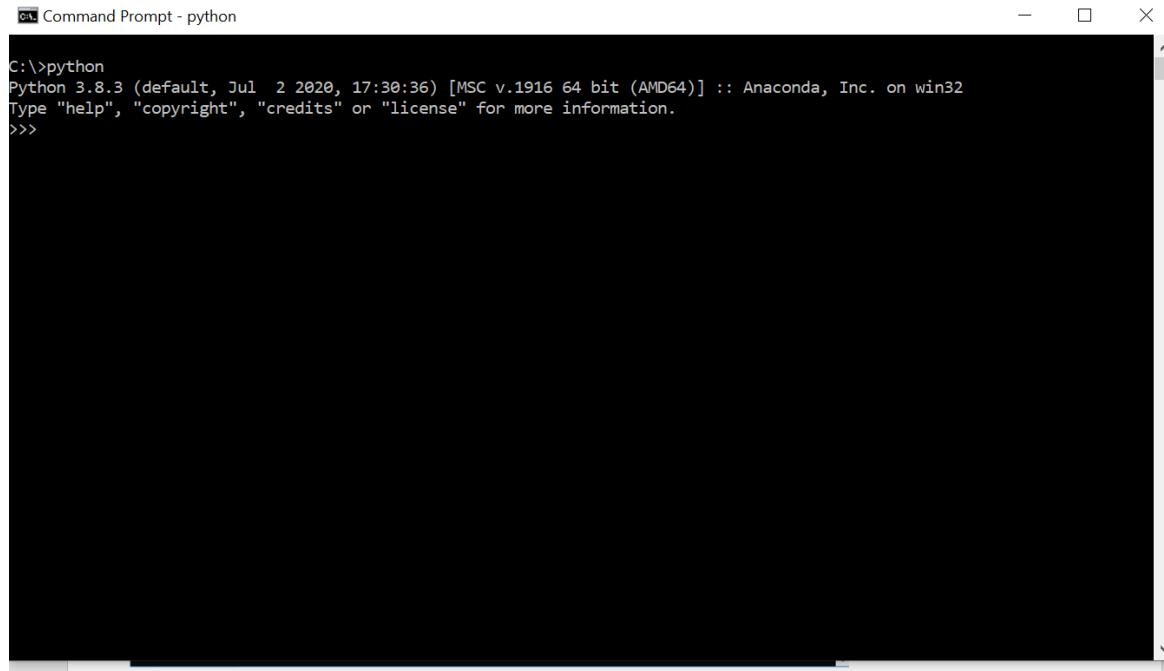
1. Go to bottom left corner of Windows where you can type
1. Enter CMD then select Command Prompt from the list



# Python Prompt via Command Prompt

To get to the Python Prompt from Command Prompt:

1. Type python at the prompt then hit ENTER
2. The >>> prompt is the Python Prompt. From here you can write and run Python code



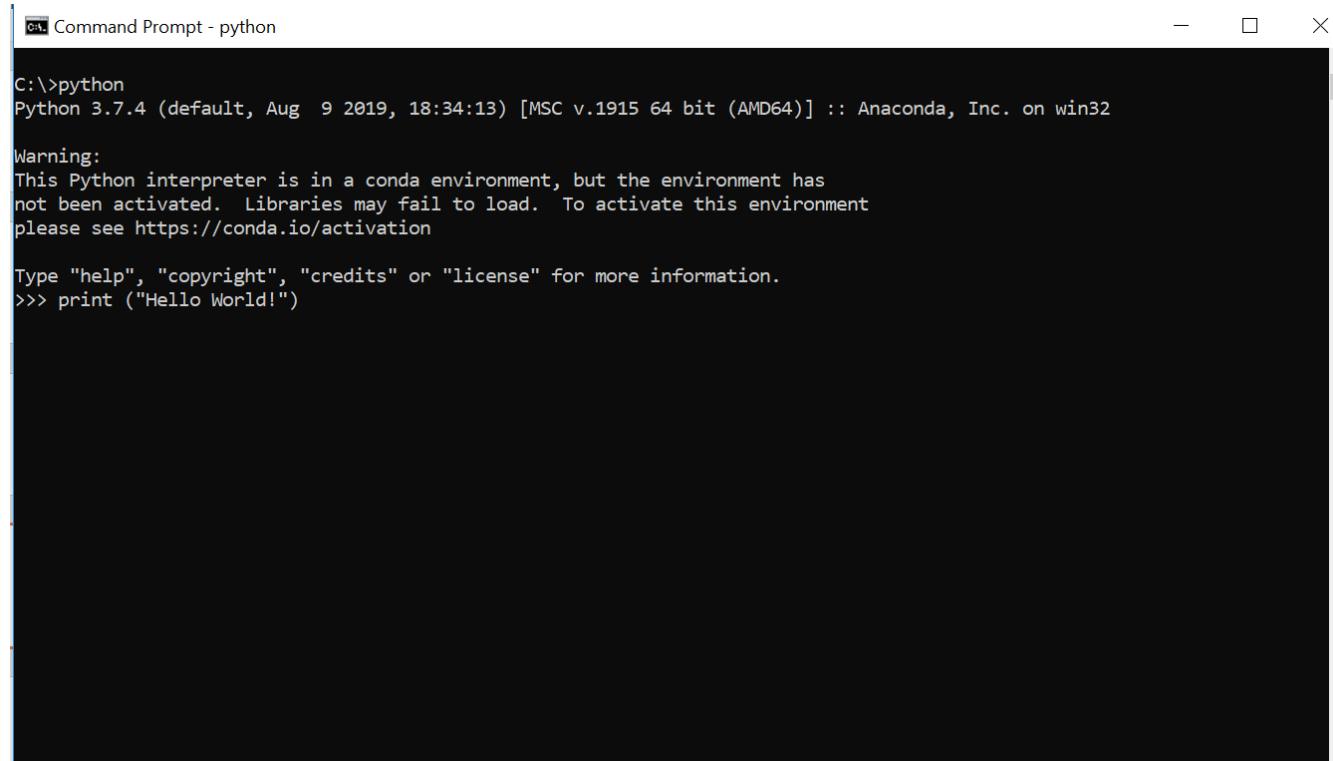
The screenshot shows a Windows Command Prompt window titled "Command Prompt - python". The window contains the following text:

```
C:\>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

# Python Prompt via Command Prompt

Let's write our first Python program via Python prompt

1. print ("Hello World!")



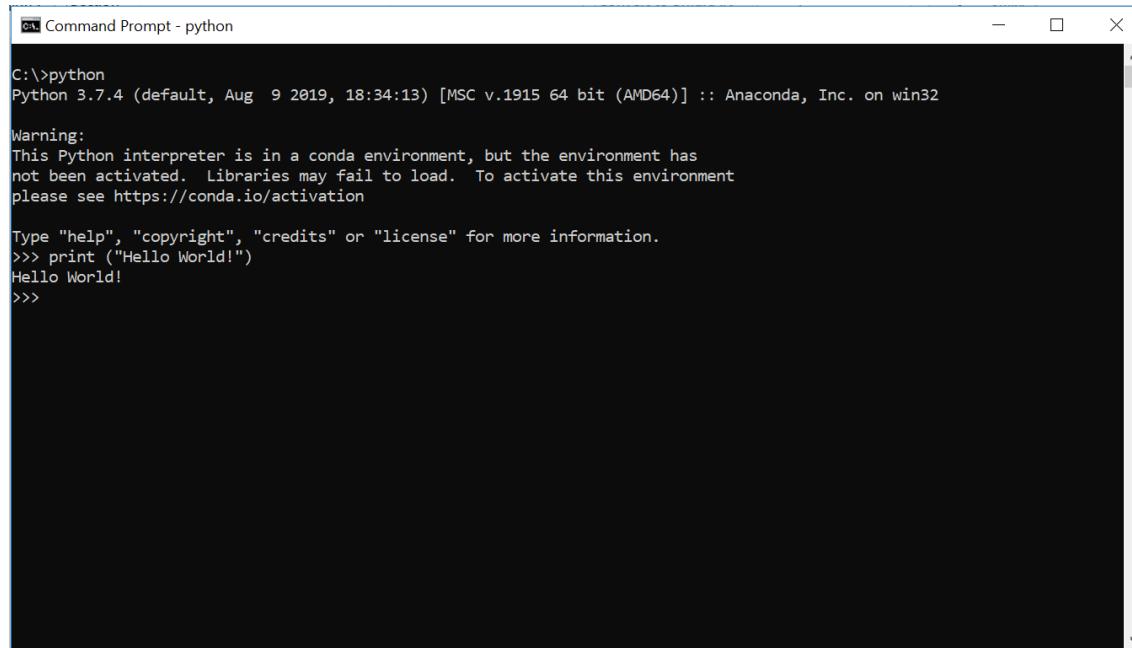
```
C:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World!")
```

# Python Prompt via Command Prompt

Let's write our first Python program via Python prompt

1. Python prompt displays (prints) your program output on the screen



```
C:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

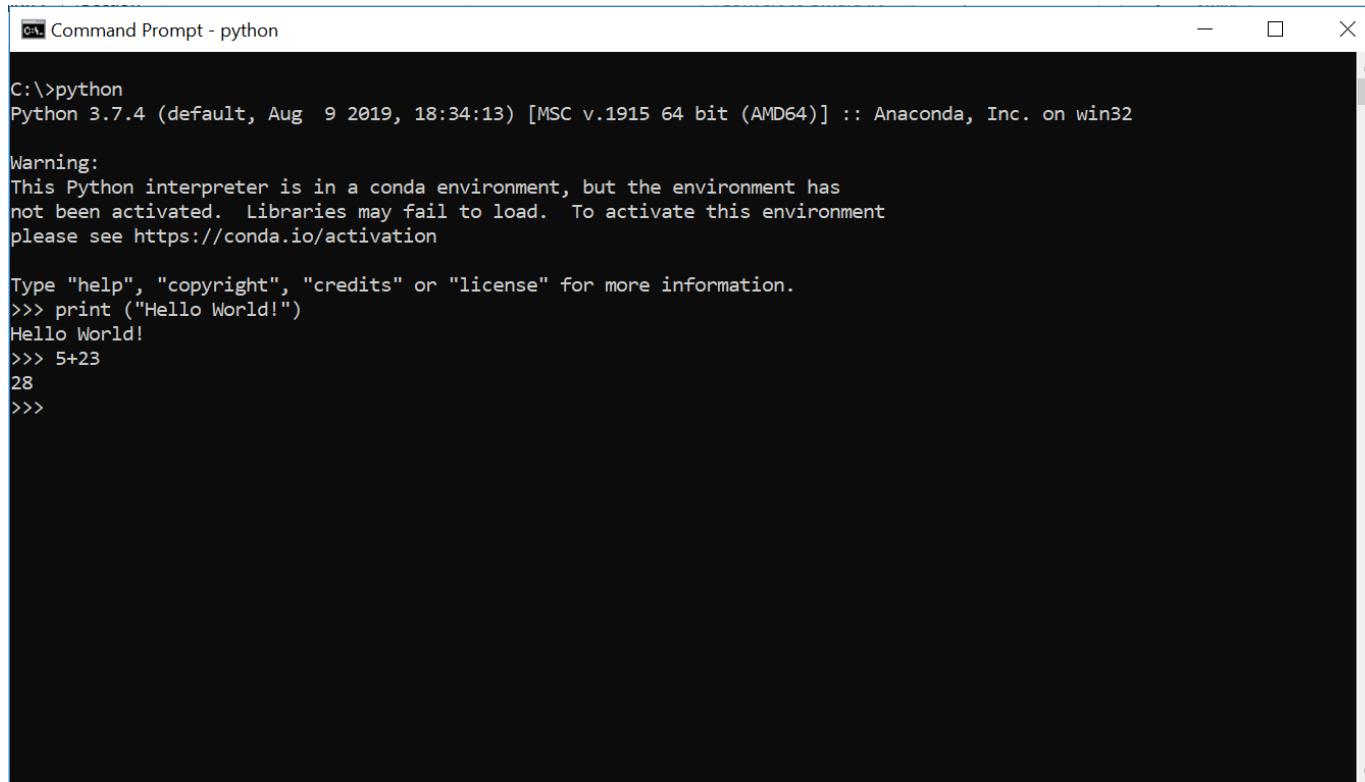
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World!")
Hello World!
>>>
```

# Python Prompt via Command Prompt

Let's use Python Prompt just like a calculator

1. Enter 5 + 23 at the Python Prompt
2. Python Prompt automatically displays the results of 5+23



```
C:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World!")
Hello World!
>>> 5+23
28
>>>
```

## Python Prompt via Command Prompt

Let's write another program that will assign values to two variables then multiply these two variables to generate result and store it in a third variable

1. Enter `x = 5` at the Python Prompt
2. Enter `y = 25` at the Python Prompt
3. Enter `z = x * y` at the Python Prompt
4. Enter `print (z)` at the Python Prompt

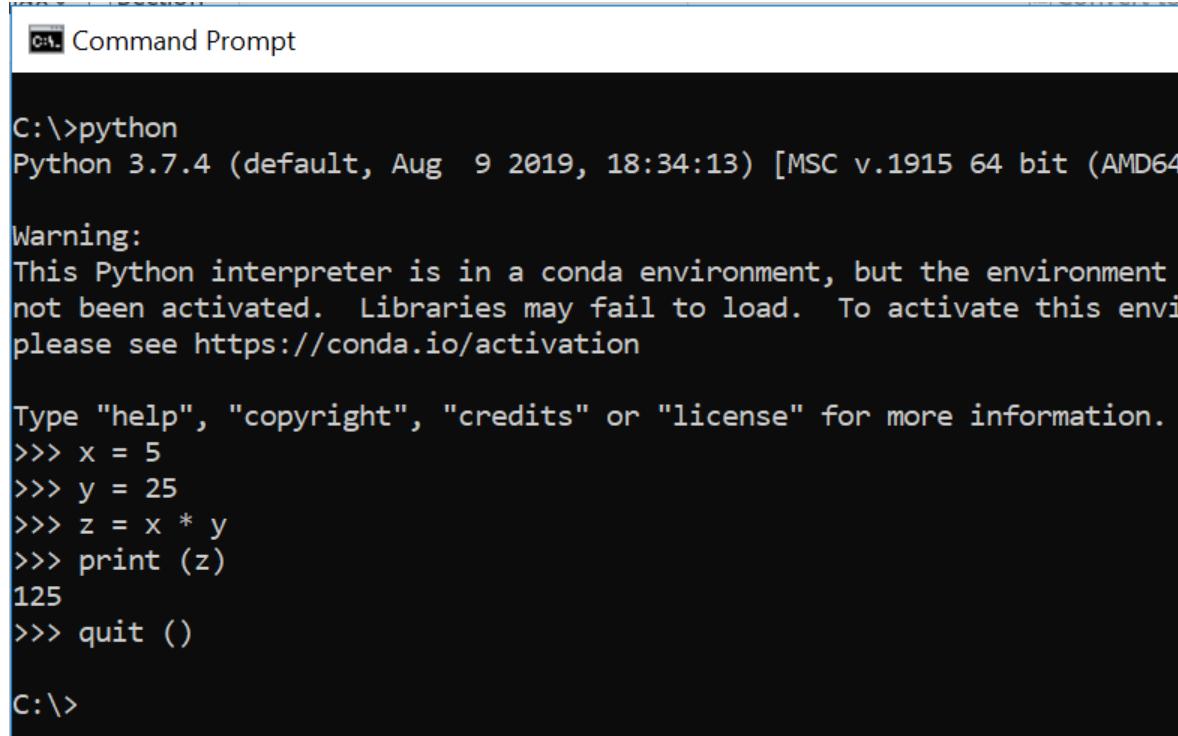
```
C:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> x = 5
>>> y = 25
>>> z = x * y
>>> print (z)
125
>>>
```

# Python Prompt via Command Prompt

To end the Python Prompt session:

1. Enter quit()



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command "C:\>python" is entered, followed by the Python interpreter's startup message for version 3.7.4. A warning message indicates that the interpreter is in a conda environment but has not been activated, noting that libraries may fail to load. The user then enters several commands: "x = 5", "y = 25", "z = x \* y", "print(z)", resulting in the output "125". Finally, the user types "quit()" to exit the session, returning to the command prompt.

```
C:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64]

Warning:
This Python interpreter is in a conda environment, but the environment
not been activated. Libraries may fail to load. To activate this envi
please see https://conda.io/activation

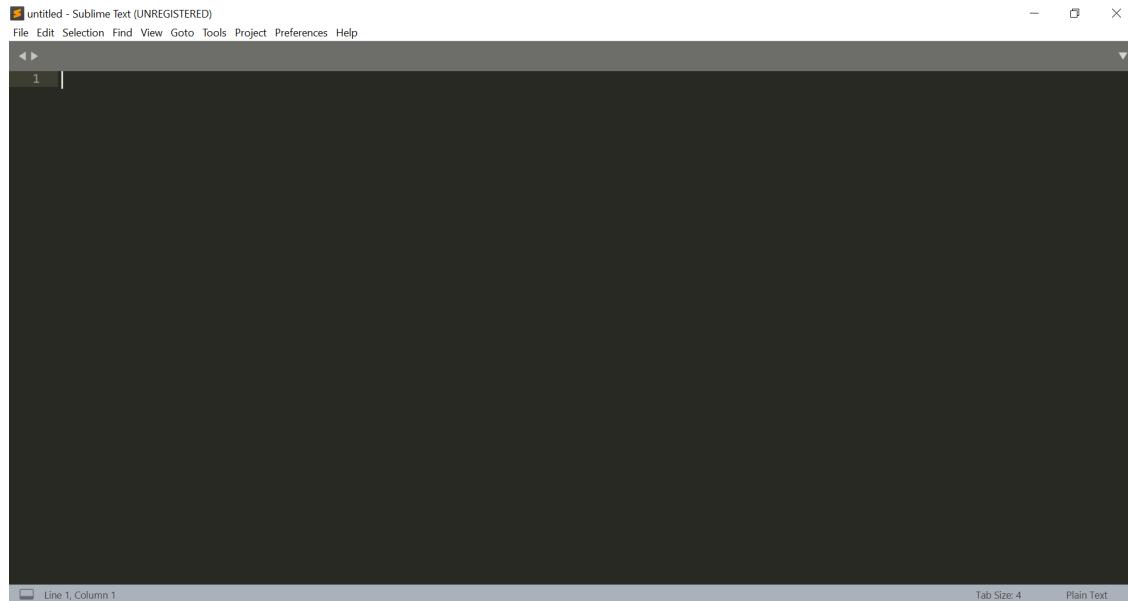
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 5
>>> y = 25
>>> z = x * y
>>> print (z)
125
>>> quit ()

C:\>
```

## Python Programming Using Editor



# Python Programming Using Editor



## Python Programming Using Editor



## Data Science: Python Programming Arithmetic Operators



## Data Science: Python Programming Arithmetic Operators

- + addition
- - subtraction
- \* multiplication
- / division
- % mod
- \*\* to the power of

My Dear Aunt Sally applies!

# Python Programming Arithmetic Operators

```
Command Prompt - python
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights

P:\>python
Python 3.7.4 (default, Aug  9 2019, 18:34:1

Warning:
This Python interpreter is in a conda environment
not been activated. Libraries may fail to
please see https://conda.io/activation

Type "help", "copyright", "credits" or "licen
>>> 1 + 1
2
>>> 1 * 2
2
>>> 1 / 2
0.5
>>> 1 - 1
0
>>> 1 % 2
1
>>> 5**2
25
>>> 5**3
125
>>>
```

# Data Science: Python Data Types



# Data Science: Python Data Types

Integer numbers	int	100
Float, real numbers	float	100.0
Text, string	str	“data science” ‘data science’
Boolean	bool	True or False

# Data Science: Python Data Types

Array		
Dictionary (unordered, changeable, indexed, no duplicates) (Can access items by [“key”])	dict	{"course": "Data Science Fundamentals", "mode": "Online", "term": "Fall 2019"} }
Lists (ordered, changeable, duplicates are ok ) (can access items by [index#])	list	["Adam", "Betsy", "Cherry", "Betsy"]
Tuples (ordered, unchangeable, duplicates ok) (can access items by [index#])	tuple	("may", "june", "july", "july", August)
Sets (unordered, unindexed, no duplicates) (cannot access items by index#)	set	{"apple", "banana", "craneberry"}

# Data Science: Python Data Types





## Python: For Loop

For loop the traditional thinking....

For counter from 1 to 100

Do.....

For Counter = 0, add 1 to the counter for each iteration until counter is less than X

## Python: For Loop

a for loop iterates over a sequence  
a list, a tuple, a dictionary, a set, or a string

For example:

```
genre = ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]
```

```
count = 0
for value in genre:
    count = count + 1
```

```
print("There are ", count, " genres of movies in the list")
```

## Python: For Loop

a for loop iterates over a sequence  
a list, a tuple, a dictionary, a set, or a string

For example:

```
genre = ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]
```

```
for value in genre:
    print(value)
```

## Python: For Loop

a for loop iterates over a sequence  
a list, a tuple, a dictionary, a set, or a string

For example:

```
genre = ["Action", "Adventure", "Animation", "Biography", "Comedy", "Crime",
"Drama", "Family", "Fantasy", "History", "Horror", "Music", "Musical", "Mystery",
"Romance", "Sci-Fi", "Sport", "Thriller", "War", "Western"]
```

```
for value in genre:
    print(value)
    if value == "Comedy":
        break
```

## Python: For Loop

a for loop iterates over a sequence  
a list, a tuple, a dictionary, a set, or a string

For example:

```
for value in 'data science':  
    print(value)
```

## Python: For Loop

Range(num1:num2)

returns numbers from num1 to num2 minus 1 (excluding num2)

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive integer exponent number")
result = 1
```

```
base = int(base)
exponent = int(exponent)
```

```
for i in range(0, exponent):
    result = result * base

print(result)
```



## Python While Loop

Loops as long as a test condition is true

# Python: For Loop

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive exponent number")
result = 1

base = int(base)
exponent = int(exponent)
```

```
for i in range(0, exponent):
    result = result * base
```

```
print(result)
```

For example:

```
base = input("Enter an integer base number")
exponent = input("Enter a positive exponent number")
result = 1

base = int(base)
exponent = int(exponent)
```

```
i = 0
while i < exponent:
    result = result * base
    i = i + 1
```

```
print(result)
```

## Python: For Loop

For example:

```
base = input("Enter an integer base number")
exponent = input ("Enter a positive integer exponent number")
result = 1
```

```
base = int(base)
exponent = int(exponent)
```

```
i = 0
while i < exponent:
    result = result * base
    i = i + 1

print(result)
```

# Data Science: Python Built-in Functions

<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

# Fruitful Function VS. Void Function (Procedure)

- Fruitful Function: returns a value to the caller
  - Int()
- Void Function (Procedure): perform tasks but not value returned to the caller
  - Print()

# User-Defined Function (Fruitful Function)

- For example: we can turn our exponent code into a fruitful function:

```
def exponent(base, exp):  
    result = 1  
  
    for i in range(0, exp):  
        result = result * base  
  
    return result
```

Save this user-defined function as MyMathFunction.py in the same folder as the main program

```
import MyMathFunction  
  
base = input("Enter an integer base number")  
exp = input("Enter a positive integer exponent number")  
  
base = int(base)  
exp = int(exp)  
  
eresult = MyMathFunction.exponent(base, exp)  
  
print(base,"^",exp," = ",eresult)
```

This is the main program. Save this main program, in the same folder as MyMathFunfion.py

# User-Defined Function (Void Function or Procedure)

```
def Compute_Mean(numbers=[0,0,0]):  
    n = len(numbers)  
  
    total = 0  
    for value in numbers:  
        total = total + value  
  
    mean = total / n  
    return (mean)
```

```
from StatMean import Compute_Mean  
  
Compute_Mean([1,1,1])
```

This is the main program.

Save this user-defined function as  
StatMean.py in the same folder as the  
main program